

Controlling mobile robot in flat environment taking into account nonlinear factors applying artificial intelligence

Tran Thi Huong, Pham Thi Thu Ha

University of Economics, Technology for Industries, Hanoi, Vietnam

Article Info

Article history:

Received Nov 11, 2023

Revised Feb 15, 2024

Accepted Mar 6, 2024

Keywords:

Actor-critic algorithm
Artificial intelligence
Autonomous navigation
Intelligent control
Mobile robot

ABSTRACT

The article shows how to build and identify intelligent automatic control problems for mobile robots in a flat surface environment at the workplace, with known and unknown obstacles. Research and develop programming and control methods as an operating system for mobile robots robot operating system (ROS). Update map data information, in the operating environment, robot position control process, obstacle overcoming process simultaneous positioning and mapping (SLAM). From there, we aim to calculate and determine the robot's motion trajectory to get a smart path. The positioning trajectory calculation system robots. The authors use actor-critic (AC) algorithm to research and develop control. Research results in simulations, in Gazebo environment and test runs on real mobile robots have shown high-quality practical performance of automatic navigation and control while using this algorithm.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Pham Thi Thu Ha
University of Economics, Technology for Industries
Hanoi, Vietnam
Email: pttha@uneti.edu.vn

1. INTRODUCTION

Currently, the automatic navigation process of each type of mobile robot can be divided into three subsystems: perception and information reception, behavioral decision making and manipulation control. On that basis, establishing a path plan is the basis for navigation and control of mobile robots [1]–[3]. The application of artificial intelligence rules has shown promise in recent years, with results achieved on a range of complex tasks such as robotic and mobile robot control, [3] and play video games in document [3], [4] from raw materials sensory input. The reinforcement learning (RL) algorithm solves these problems by learning a policy that maximizes the reward function considered as part of the problem formulation. There is little practical guidance provided in RL theory on how to design, these rewards. However, with an action environment intended to control a robot, it is often practiced in finite state form (applied with Markov-Decision Process (MDP) decision process) and the use of artificial intelligence-RL algorithms for the context. These are dynamic planning techniques that have a high level of quality coverage. The process of researching, calculating, selecting, and setting motion probabilities and probabilities to achieve in MDP is usually a random but static factor in the process of implementing the robot tracking control problem. This process is not at all like supervised learning, with RL, there is no exact (Input–Output) pair data and some sub-optimal spatial actions have not been clearly evaluated as correct or not. is wrong. Furthermore, the performance of online learning, updating, and perception processes is of great interest here: including data searching, parametric trade-offs between exploration (mapping of uncomputed territory) and fully exploit (existing knowledge). There are generally two methods, commonly used to solve decision problems: searching the strategy space and searching the space of value functions, also known as “strategy iteration” and “strategy iteration”, “duplicate

values". These two methods are to study the use of feature reinforcement learning algorithms. In addition, with recent research, some scientists have proposed a method that combines the above two methods, which is the actor-critical learning method [4]–[6].

Up to now, we have worked a lot on classical (old) and traditional (not very effective) controllers such as: using proportional-integral derivative, fuzzy PD, method PD+I, PI, linear quadratic regulator (LQR) and many other classical controllers, [1]–[5]. Most profound problem with those methods is that they do not (yet) bring any benefit, these controllers are often adjusted manually (the quality is not up to today's controls). Therefore, it always causes many errors (noise, nonlinear factors) and the system does not work correctly, causing quality of the controller to not be high. Working system is not optimal, not synchronized, not good. Many times optimal value cannot be achieved but only stops at the simulation level. Biggest benefit of the RL algorithm as a controller is that the model self-adjusts the function and algorithm to achieve optimal values such as: Q-learning, DQN algorithm, precise policy gradient algorithm deep identification (DDPG), state action status bonus (SARSA), and PG. There are two main and basic RL methods; one is a value function based method, and the other is a policy function based method, implemented in the action space. The AC algorithm is also the basis for accessing modern RL algorithms such as advantage agent criticism (A2C), asynchronous advantage agent criticism (A3C), this is an algorithm that brings many benefits. will be used in the future. many [4], [7]–[11].

In paper the authors present a self-learning algorithm, based on RL rules in automatic control, practice and experimentation in cases where the environment is not specifically determined. Experimental studies, to be conducted on navigation take for robots. More specifically, we introduce the AC method as a gradient learning algorithm that implements a policy function, learning both the policy, acting as an agent, and state value function as a critic for generalization, and estimate values of all states of the RL algorithm, [3], [10]–[13]. The AC algorithm solves problem of finding paths and avoiding obstacles (fixed obstacles, moving obstacles, etc...) for robots in real, unknown nonlinear working environments and action spaces.

2. RESEARCH AND BUILD A MOBILE ROBOT CONTROL MODEL WHICH TAKES NONLINEAR FACTORS INTO ACCOUNT

The kinematics and dynamics of a mobile robot with a three-wheeled model with the structure described in Figure 1. Two coordinate systems can be used to describe movement of the robot. One is the global coordinate frame (x, y) defined and the other is local coordinate frame (left, right, etc...) defined on the robots. When the coordinates are chosen in Figure 1, it is robot's velocity contains: linear velocity along the Ox axis, angular velocity along the Oy axis. And includes the control block and functional blocks as follows: the robot frame is designed in the style of a three-wheeled robot; with a distance between two wheels of 0.35 m and a wheel radius of 0.065 m.

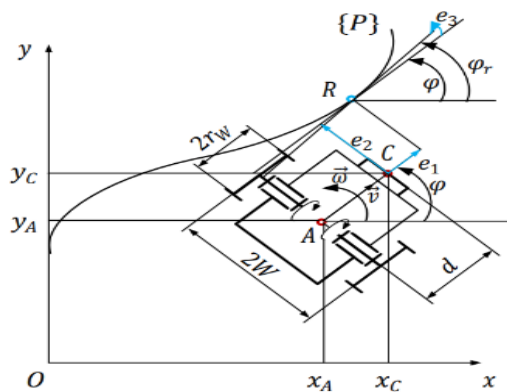


Figure 1. The model implements the path planning motion of a mobile robot

Kinematic equation of the driving device of the two main motors of the robot written at point A is the midpoint of the straight line connecting the centers of the two driving wheels:

$$v = v_A = \frac{(\omega_R - \omega_L)r_W}{2} \quad (1)$$

$$\omega = \frac{(\omega_R - \omega_L)r_W}{2W} \quad (2)$$

From (1) and (2), we have the equations of motion Robot written in coordinate system at point A, point C (the center of the robot is where the path sensor is located):

$$\begin{bmatrix} \dot{x}_A \\ \dot{y}_A \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos \varphi & 0 \\ \sin \varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \dot{x}_A - \dot{\varphi} d \sin \varphi \\ \dot{y}_A - \dot{\varphi} d \cos \varphi \\ \omega \end{bmatrix} \quad (4)$$

The robot path tracking controller is always implemented according to Lyapunov's stability standards [1]–[3]. Although it responds well in simulation, it is limited in practice because the path tracking sensor can only measure horizontal deviation (or e_2). Therefore, the article presents a controller designed in the form of linear feedback [2], [5], [6], [13]. Therefore, we will calculate according to the equation of tracking error as:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} v_r \cos e_3 \\ v_r \sin e_3 \\ \omega_r \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -d - e_1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (5)$$

When the mobile robot follows the moving trajectory, only vibrations occur around the robot; Therefore, the equilibrium point of the nonlinear system in (5) is $X_0 = [e_{10} \ e_{20} \ e_{30}]^T = [0 \ 0 \ 0]^T$ with input $u_{sk0} = [v_r \ \omega_r]^T$. Performing linearization (5) around the equilibrium point, we get:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_r & -v_r \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -d \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (6)$$

In reality, the speed of the mobile device during operation is constant, so $e_1 \approx 0$, then, (6) becomes:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & v_r \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} -d \\ -1 \end{bmatrix} u, u = \omega \quad (7)$$

Differentiating the first equation of (7) and substituting the second equation we get:

$$\ddot{e}_2 = -d \times \dot{u} - v_r \times u \quad (8)$$

Set: $x_1 = e_2$, $x_2 = \dot{x}_1 - \beta \cdot u$ with $\beta = -d$, ta có:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -d \\ -v_r \end{bmatrix} \cdot u \quad (9)$$

In (9) has the form $\dot{X} = A \cdot X + B \cdot U$ and the controllable matrix $M = [B \ AB]$ has $\det(M) \neq 0 \ \forall \ v_r \neq 0$, so system (9) is a controllable system. Set the feedback control law $u = -K \cdot X$ with $K = [k_1 \ k_2]$. Expanding the control law we get:

$$u = \frac{-k_1}{1+k_2d} e_2 + \frac{-k_2}{1+k_2d} \dot{e}_2 \quad (10)$$

The control law (10) has the following form: $u = K_P e_2 + K_D \cdot \dot{e}_2$ với $K_P = \frac{-k_1}{1+k_2d}$, $K_D = \frac{-k_2}{1+k_2d}$. The coefficients k_1, k_2 are determined based on the LQR calculation method. With the instantaneous center velocities of the motion drive system and the mobile robot at time (i) are $C_{O(i)}$ and $C_{A(i)}$ respectively. To determine the calculation of the deflection angle between the motion transmitter and the robot at time (i) is:

$$\theta_{(i)} = \varphi_{(i)} - \varphi_{A(i-1)} \quad (11)$$

Because the motion transmitter and the robot are linked together through the hinge joint at O so $v_D = v_A = v_O = v$, the radius of rotation and angular velocity of mobile robot at time (i) are calculated as:

$$C_A O_{(i)} = \frac{L}{\sin(\theta_{(i)})} \quad (12)$$

$$\omega_{A(i)} = \frac{v_{(i)}}{C_A O_{(i)}} = \frac{v_{(i)} \sin(\theta_{(i)})}{L} \quad (13)$$

In fact, to describe and build a dynamic model for an omnidirectional mobile robot that complies with the control program, it is necessary to have a solid motion transmission mechanism; combined with data processing: identification equipment such as intelligent cameras. Sensors and intelligent control algorithms, control programs for robots to follow certain orbits in space and working environment according to parameter $e_{x,k} \approx 0$, $e_{y,k} \approx 0$, $e_{\theta,k} \approx 0$, with k being a constant. From there, this mobile robot will perform cognitive and identification processes with the working environment, to perform smooth automatic navigation: avoiding fixed, moving obstacles, etc... to reach the destination safely and perfectly, [4], [5], [7], [14], [15].

3. DEVELOPMENTAL AC ALGORITHM APPLICATION FOR MOBILE ROBOTS TAKING INTO ACCOUNT NONLINEAR FACTORS

3.1. Model of the AC algorithm

In RL methods, such as the policy gradient-PG algorithm, the AC algorithm is a policy artificial intelligence algorithm with many outstanding advantages, learning online, not depending on model free model with the method new combinations when using the algorithm. The methods of the AC algorithm allow to combine many advantages of the PG algorithm and value function (VF) method. This AC algorithm also focuses on solving optimization and random problems: related to probability distribution, etc..., this is the advantage of this algorithm.

Here, we focus on researching the problem of controlling robots in general and mobile robots in particular, in cumulative reward. With main idea of this algorithm, build a model from two action space components. Here we take as input estimated state S and output is distribution of each action in set (entire) data action space. This element belongs to the group of methods based on policy functions, it plays the role of learning optimal policies. Remaining element is critique network of value-based methods. Component behaves as a function of $V(s)$ values, so computing the resulting estimate (the expectation of the long-term cumulative reward) is a matter of determination. From value function, can build certain assessments of control system, thereby updating control-related parameters, [4], [12], [16]-[20].

The execution sequence of the AC algorithm, according to Figure 2 is as: the first is to randomly initialize model parameter θ_μ for network Actor $\mu(s)$. Performed to initialize random parameters in terms of model parameters θ_v for network critic $V(s)$. The second step, for each step of the training process, the algorithm is repeated as follows: using the existing policy function $\mu(s)$ produces N consecutive samples: $s_{ts}, a_{ts}, R_{ts+1}, \dots, \theta_{ts+N-1}, a_{ts+N-1}, R_{ts+N}, S_{ts+N}$. At start training, $ts=1$. Here for subsequent trainings in same training set, assign $ts \leftarrow ts + N$. Parameter N is set by “NumStepsToLookAhead” argument in “rlACAgentOptions”, [3], [7], [21], [22].

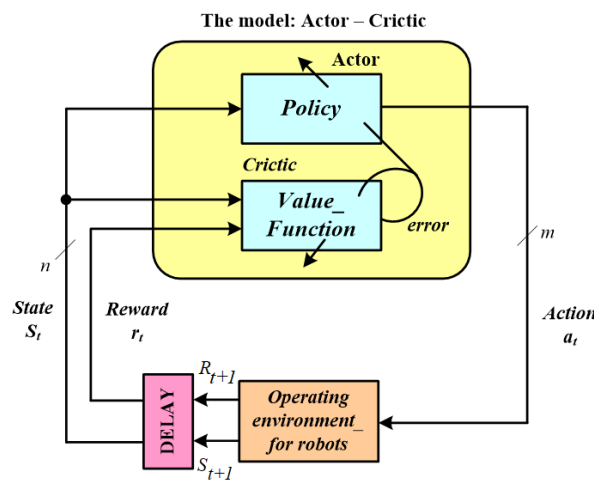


Figure 2. Process diagram of implementing the AC algorithm

The third is to compute the AC algorithm with state in N (samples), $(G_{ts+1}, \dots, G_{ts+N})$.

$$G_t = \sum_{k=t}^{ts+N} \gamma^{k-t} R_k + b\gamma^{N-t+1} V(S_{ts+N} | \theta_V) N = P(S_{t+1} | S_t, a_t) \quad (14)$$

In (14) is done with: $\forall t = ts + 1, ts + 2, \dots, ts + N$ and γ (is considered the discount factor). Next, we calculate profit margin function for position above:

$$\delta_t = G_t - V(S_t), \text{ when } \forall t = ts + 1, ts + 2, \dots, ts + N \quad (15)$$

Perform cumulative gradient Actor network on N samples, with μ parameterised actor:

$$\nabla_{\theta_\mu} J = \sum_{t=1}^N \ln \mu(S_t) \delta_t \quad (16)$$

The cumulative gradient for the critical with G_t and $V(S_t)$:

$$\nabla_{\theta_V} J = \sum_{t=1}^N \nabla_{\theta_V} (G_t - V(S_t))^2 \quad (17)$$

Here we need to pay attention to implementing the “Entropy Loss Weight” factor when programming with algorithm. Next is to update all network parameters:

$$\theta_\mu = \theta_\mu + \alpha \cdot \nabla_{\theta_\mu} J \quad (18)$$

$$\theta_V = \theta_V + \beta \cdot \nabla_{\theta_V} J \quad (19)$$

In which, α and β are learning AC algorithm. Here we take the final step to stop the loop (calculation has been optimized) when the algorithm has converged.

Furthermore, the critic has a computationally time-varying error pattern that learns and critic whatever policy the agent is currently following of the algorithm. Those estimates are temporary difference errors:

$$\delta_t = r_{t+1} \cdot V(S_{t+1}) - V(S_t) \quad (20)$$

Suppose actor is implemented by a stochastic policy $\pi(a|s, \theta)$ parameterized by vector θ . Critic's error estimate δ_t allows and according to:

$$\theta_{t+1} = \theta_t + \alpha \cdot \delta_t \frac{d \log \pi(a_t | s_t, \theta_t)}{d \theta_t} \quad (21)$$

The α component is considered learning rate of agents. The critic will have its value function updated according to the general temporal difference procedure given by:

$$V(S_t) = V(S_t) + \beta \cdot \delta_t \quad (22)$$

In robotics, using the AC algorithm to deploy action agents to intelligently automatically navigate the mobile robot to follow moving trajectories, avoid dynamic obstacles, as well as static the obstacles in moving process of robot. The robot also calculates shortest trajectory for robots to move to the destination with fastest path [22]-[25].

3.2. AC algorithm applies intelligent navigation to mobile robots

When studying RL algorithms, we see that algorithms here are rarely mutually exclusive, most of these algorithms always complement and support other algorithm. In particular, we always have PG algorithm and the AC algorithm. Since then, attention brought to this approach has greatly increased its efficiency, mainly because they have two clear advantages: they require minimal computation to select actions (when considering nonlinear components, etc.) and they can learn an explicit random policy (related to the probability distribution problem) to bring high control results. The results will be more accurate than other algorithms, [11], [13], [20], [22]-[24].

When using the AC algorithm (algorithm allowed for online learning) it is always updated more frequently according to time value differential learning method. This problem means that system parameters will be updated immediately after one iteration, several iterations (here number of steps depends on installation and programming process, usually suggested to allow is $N=\{32, 64, 128, \dots\}$). The research

method will contribute to minimizing errors in system. Process of implementing navigation determination robots, the ability to deliver results from learning data will also increase significantly, [5], [9], [10], [21]. Furthermore, implementing the calculated value function $V(s)$ with a reasonable design and choice brings high benefits. This will play an important role in determining the optimal parameter value when learning, which will then directly participate in calculating profit function. We can see AC algorithm below implementation of navigation for robots, [13], [16], [20], [25]-[27]. The procedure and process of making AC algorithm are summarized in Algorithm 1.

Algorithm 1. The actor-critic algorithm for robot

```

1:   Initialize: Initialize policy  $\pi(a | s, \theta)$  with initial parameter  $\theta = \theta_0$ , its
2:   Derivative  $d \log \pi(a | s, \theta)$ , an arbitrary value function  $V^{\pi}(s)$  and set the
   eligibility trace  $z((\theta)) = 0$ .
3:   Repeat until  $\pi(a | s, \theta)$  is optimal:
       a)  $a_t \leftarrow$  action given by  $\pi(a_t | s_t, \theta_t)$ 
       c) Take action with  $a_t$ , observe next state  $s_{t+1}$  and reward  $r_{t+1}$ 
4:   Critic evaluation and update parameter:
7:       b)  $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ 
8:       e)  $V(s_t) = V(s_t) + \alpha \delta_t$ 
9:   Actor update:
10:      c)  $z_{t+1}(\theta) = \lambda \cdot z_t(\theta) + \frac{d \log \pi(a_t | s_t, \theta_t)}{d\theta_t}$ 
11:      d)  $\theta_{t+1} = \theta_t + \alpha \delta_t z_{t+1}(\theta)$ 
12:   Clean out the local buffer
13:    $s_t \leftarrow s_t + 1$ 
14:   end
15:   end

```

Then, the AC algorithm is implemented; with action agents to realize automatic navigation for robots. This process allows the robot to perform movement trajectories to avoid dynamic and static obstacles during movement of robot. Then, combined with the robot's control system (smart sensors, smart cameras, and power transmission system) it also allows calculating the shortest trajectory for robot to move to destination with a fast path most [3], [10], [14]. The specific implementation process of AC algorithm to navigate robots in environment taking into account nonlinear factors is described in detail above.

4. RESULTS AND DISCUSSION OF RESEARCH PROBLEMS

Here the authors research a model of autonomous the robots in a flat the environment taking into account nonlinear factors. This mobile robot with its actual hardware architecture includes: 3600 LiDAR system with SLAM and navigation, Scalable architecture, Single with board main computer as (Raspberry Pi 5 the Broadcom BCM2712), DYNAMIXEL x 2 for Wheels, include sprocket Wheels for Tire, Li-Po Battery 11.1 V 2,800 mAh 45 C; are designed in full detail with perfect robot structure; with hardware module perform a number of tasks, according to order operation this mobile robots: such as finding a path, overcoming obstacles, automatically navigating, and finding the end point.

In the experiments, the authors performed several tasks according to the mobile robot's operating sequence; with perfect hardware structure. For example, the Raspberry Pi 5 running Ubuntu has an Arm Cortex A76 Broadcom BCM2712 @ 2.4 GHz dual-core processor, making it up to three times faster than previous generation mobile robots. Raspberry Pi 5 Model BCM2712 embedded computer directly processes information from a series of intelligent devices including: sensors, cameras, Astra, etc... then transmits commands to the intelligent microcontroller.

To do this when the robot records images from the environment, as well as measures the distance between the robot unknown obstacles (taking into account nonlinear factors), the mobile robot is equipped with cameras and smart sensors, including intelligent cameras can laser scan 360 degrees and range within 26 m to create map data for use in work mapping process, intelligent microprocessor control circuit system will be part that receives control signals of Raspberry Pi 5 and then sends signals to MOSFET bridge circuit to operate two main motors of robot perfectly.

Environment to perform robot navigation with depth camera and randomly placed obstacles in a realistic environment (with many unknown obstacles). This is mainly a visualization tool, which can provide live updates of the maps that have been created from the AC algorithm built as above. The robot's working trajectory on map can also be displayed in real environment. Here, training and data recognition take place so

that the robot knows when to avoid (unknown) obstacles while moving safely and flawlessly to its destination so that it cannot collide.

Figure 3 shows the map built model on Gazebo, created map has rectangular obstacles, circular cylinders and robots (Green), with intelligent cameras measuring depth and other objects. Obstacles will be placed randomly in Gazebo environment in Figure 3(a). A mobile robot that is guided and carries out instructions will automatically move around environment to retrieve necessary data (calculations, processing,...) that will be performed to build a map (set paths, routing). Purple line going to yellow dot is considered a laser scanning signal, this is generated from RPLidar and robot's current position is updated by geometric measurements to remove obstacles, Figure 3(b).

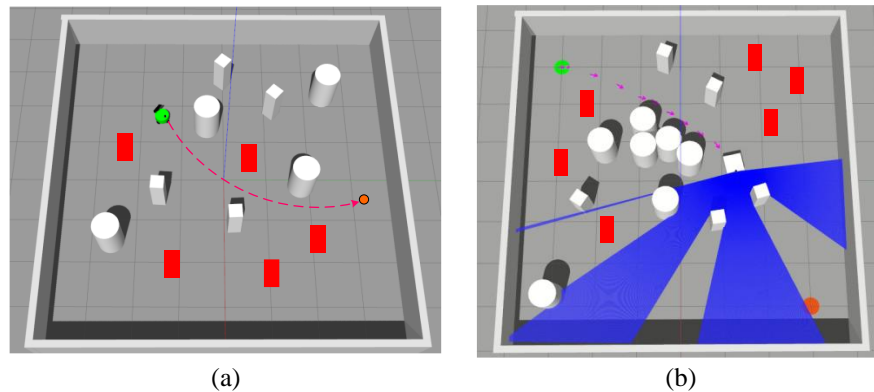


Figure 3. Simulation results: (a) build a visual map and robot model on the Gazebo environment when avoiding obstacles and finding the shortest path and (b) build maps and robot paths in the Gazebo when performing detailed work with a 360-degree scanning camera.

Figure 4 presents the learning results in the simulation environment with the robot operating system and MATLAB Simulink. We can see that average reward (real reward) of robot in each teaching episode continuously increases as the training process continues (from low to high). The robot will learn knowledge about the environment through interacting with the environment on the basis of the AC algorithm (input is the observed state, output is probability of each action occurring in selected set of values). The authors have researched above. The teaching set value (performed up to 1,000 sets) corresponds to the reward in increasing order, Figure 4. Finally, the robot can navigate to its destination quickly (even in the presence of nonlinear factors) and autonomously in both simple and complex environments without any collisions with other objects obstacle.

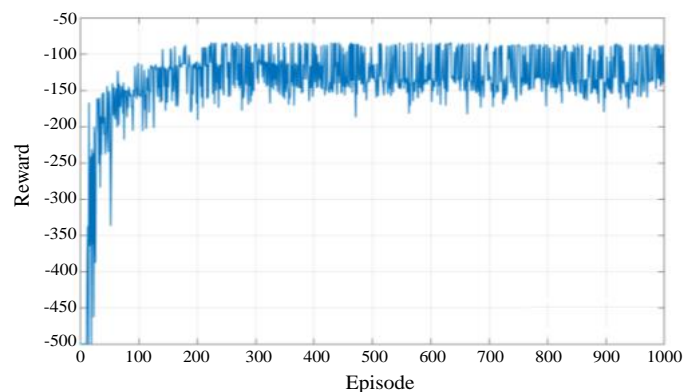


Figure 4. Results of the overall reward study with training in a flat environment considering nonlinearity

Experimental research shows that the effectiveness of AC algorithm model proposed by the author always ensures that it meets the set requirements and satisfies the control process well. The Actor using

network takes current state of built robot as input with a maximum linear velocity of 0.35 m/s and a maximum with angular velocity of 3.5 rad/s (200.75 degrees/s). For Critic, current state, action space values are calculated for prediction, when using only AC algorithm directly connected to process the data parameters with the input state and input data out to the unified system robot control system. In which the actions and outputs of AC network are connected on basis of a deterministic reinforcement learning algorithm.

The research results show that research and development of new control algorithms, combined with simulation tools to verify robots, is extremely necessary. Experimental methods, MATLAB tools, ROS operating system, simultaneous mapping built on Gazebo and in Rviz. This visualization tool that can live updates of maps created from the SLAM algorithm to control mobile robots. Furthermore, the robot's movement trajectory in map can also be automatically navigated, and this environment obstacles can always be created, as shown in Figure 3, for the robot to move (even in the environment nonlinear). The results show that the robot has established a motion trajectory and moved to the desired target accurately and reached the destination safely. These results bring high practical benefits, which can be performed on most tasks when controlling robots in industrial factories, mobile robots in transportation, autonomous robot in healthcare and autonomous robot in healthcare factories in Viet Nam and in factories the world.

5. CONCLUSION

The content of the article presents the control of a three-wheeled self-propelled robot using intelligent navigation in an unknown flat environment (taking into account nonlinear factors), using simulation tools for control programming. Simulation results in gazebo software demonstrates ability of an autonomous robot to automatically, intelligently navigate to desired target locations (destination) and avoid static and dynamic obstacles while moving along path in a simple environment simple and complex. This study has shown the practical effectiveness of the robot control process and automatic navigation for the robot that the author has researched; moreover, this result shows that the robot has built a motion trajectory, moved to the correct target, and automatically avoided moving obstacles that appeared on the path without encountering any obstacles. The development direction of the research problem is expected to be applied to a number of practical self-propelled robots in industrial factories, in civil works, in smart transportation and in medicine with the more optimized algorithms of machine learning.

ACKNOWLEDGEMENTS

This research was supported by Faculty of Electronic Engineering Technology and computer engineering; University of Economics-Technology for Industries; No. 456 Minh Khai Road, Hai Ba Trung district, Hanoi Capital-Vietnam National; <http://www.uneti.edu.vn/>.





REFERENCES

- [1] Mohit Sewak, "Deep Reinforcement Learning", Frontiers of Artificial Intelligence Springer Nature, 2019.].
- [2] N. T. Tuan, "Base deep learning", *The Legrand Orange Book. Version, Version 2, Last Update*, 2020.
- [3] V. T. T. Nga, O. X. Loc, and T. H. Nam, "Enhanced learning in automatic control with Matlab simulink". Hanoi Polytechnic Publishing House.
- [4] C. C. Aggarwal, "Neural networks and deep learning". Cham: Springer International Publishing, 2018, doi: 10.1007/978-3-319-94463-0.
- [5] A. Bacciotti, "Stability and control of linear systems," vol. 185. Cham: Springer International Publishing, 2019, doi: 10.1007/978-3-030-02405-5.
- [6] M. Jufer, *Electric drives*. Wiley, 2013, doi: 10.1002/9781118622735.
- [7] J. Connor and S. Laflamme, "Advanced control theory," in *Structural Motion Engineering*, Cham: Springer International Publishing, 2014, pp. 545–599, doi: 10.1007/978-3-319-06281-5_10.
- [8] N. P. Quang and J.-A. Dittrich, *Vector control of three-phase AC machines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, doi: 10.1007/978-3-662-46915-6.
- [9] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against IoT-connected home environments," *Procedia Computer Science*, vol. 134, pp. 458–463, 2018, doi: 10.1016/j.procs.2018.07.183.
- [10] X. Ruan, D. Ren, X. Zhu, and J. Huang, "Mobile robot navigation based on deep reinforcement learning," in *2019 Chinese Control And Decision Conference (CCDC)*, Jun. 2019, pp. 6174–6178, doi: 10.1109/CCDC.2019.8832393.
- [11] T. D. Chuyen, D. H. Du, N. D. Dien, R. V. Hoa, and N. V. Toan, "Building intelligent navigation system for mobile robots based on the actor-critic algorithm," *Lecture Notes in Networks and Systems*, pp. 227–238, 2022, doi: 10.1007/978-3-030-92574-1_24.
- [12] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *Robotics: Science and Systems*, 2019, doi: 10.15607/RSS.2019.XV.073.
- [13] N. T. T. Vu, L. X. Ong, N. H. Trinh, and S. T. H. Pham, "Robust adaptive controller for wheel mobile robot with disturbances and wheel slips," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 1, pp. 336–346, 2021, doi: 10.11591/ijece.v11i1.pp336-346.





- [14] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 79–86, doi: 10.1109/IROS.2017.8202141.
- [15] J. Pyrhönen, V. Hrabovcová, and S. Semken, *Electrical machine drives control: an introduction*. Wiley, 2016, doi: 10.1002/9781119260479.
- [16] K. Zhao *et al.*, "System informatics: from methodology to applications," *IEEE Intelligent Systems*, vol. 30, no. 6, pp. 12–29, 2015, doi: 10.1109/MIS.2015.111.
- [17] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *arXiv*, Apr. 2019, doi: 10.48550/arXiv.1904.07854.
- [18] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-September, pp. 31–36, 2017, doi: 10.1109/IROS.2017.8202134.
- [19] T. T. Pham, M. T. Le, and C.-N. Nguyen, "Omnidirectional mobile robot trajectory tracking control with diversity of inputs," *International Journal of Mechanical Engineering and Robotics Research*, pp. 639–644, 2021, doi: 10.18178/ijmerr.10.11.639-644.
- [20] M. Hijikata, R. Miyagusuku, and K. Ozaki, "Wheel arrangement of four omni wheel mobile robot for compactness," *Applied Sciences*, vol. 12, no. 12, p. 5798, Jun. 2022, doi: 10.3390/app12125798.
- [21] V. T. T. Linh *et al.*, "The orbit tracking adaptive control for omnidirectional mobile robot based on RBF neural network," *Conferences Advances in Engineering Research and Application*, 2023, pp. 952–967, doi: 10.1007/978-3-031-22200-9_101.
- [22] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," *arXiv*, Oct. 2016, doi: 10.48550/arXiv.1610.00633.
- [23] E. Clotet and J. Palacín, "SLAMICP Library: accelerating obstacle detection in mobile robot navigation via outlier monitoring following ICP localization," *Sensors*, vol. 23, no. 15, Aug. 2023, doi: 10.3390/s23156841.
- [24] W. Na, Y. Lee, N.-N. Dao, D. N. Vu, A. Masood, and S. Cho, "Directional link scheduling for real-time data processing in smart manufacturing system," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3661–3671, Oct. 2018, doi: 10.1109/IIOT.2018.2865756.
- [25] S. Du, Y. Li, X. Li, and M. Wu, "LiDAR odometry and mapping based on semantic information for outdoor environment," *Remote Sensing*, vol. 13, no. 15, Jul. 2021, doi: 10.3390/rs13152864.
- [26] D. Griffiths and J. Boehm, "A review on deep learning techniques for 3d sensed data classification," *Remote Sensing*, vol. 11, no. 12, Jun. 2019, doi: 10.3390/rs11121499.
- [27] "ROS tutorials." <http://wiki.ros.org/ROS/Tutorials> (accessed Oct. 28, 2023).

BIOGRAPHIES OF AUTHORS



Tran Thi Huong     was born in Nam Dinh. She graduated in Electronic Engineering from Le Quy Don University of Technology, Hanoi, Vietnam in 2009. Currently working at the Faculty of Electronics, University of Economics-Industrial Technology. Her research interests are electronic engineering, telecommunications engineering, information technology, automated robots, artificial intelligence, control applications, and neron control networks. She can be contacted at email: huongtt@uneti.edu.vn.



Pham Thi Thu Ha     was born in Nam Dinh. She graduated with a master's degree in Electronic Education from the University of Technology, Hanoi, Vietnam in 2010. Currently working at the Faculty of Electronics, University of Economics-Industrial Technology. Her research interests are electronic engineering, electronics applied in industry, wireless sensor network, internet of things (IoT), artificial intelligence (AI), and telecommunications network. She can be contacted at email: pttha@uneti.edu.vn.